Testing of GALS Components

Deepali Basavaraj Koppad, PhD
PES Institute of Technology
ECE Department, 100 Feet Road
Bangalore, India 560085

ABSTRACT
The purpose of this paper is to perform structural testing on GALS (Globally Asynchronous and Locally Synchronous) components, such as wrapper designs. GALS consists of three main parts: synchronous block, I/O ports and a local clock generator. The I/O Ports and the local clock generator form the wrapper design. Testing has been done for every net in each of the wrapper components. The feedback nets that are usually uncontrollable are also tested using same methodology. The collapse ratio and the maximum number of test vectors required are calculated for every component. A 2:1 mux is used to detect 3 faults that could not be detected using structural testing. Fault coverage of 100% is obtained for every component of the wrappers. The testing is performed on two different wrappers using Pyxis schematic in Mentor Graphics for 180 nm technology.

General Terms
Structural Testing

Keywords
GALS, Wrapper Testing, Fault coverage.

1. INTRODUCTION
The synchronous circuits have prominent disadvantages of clock skew and power consumption due to global distribution of clock. The extra overhead arises due to synchronizers used in cases when multiple clock frequencies are required on a single chip. Hence, researchers are exploring alternatives such as, NULL Convention Logic (NCL), Asynchronous Circuits, Globally Asynchronous and Locally Synchronous (GALS) design styles. The advantage of NCL and Asynchronous design styles is that they do not use the clock signal for synchronization. Infact they make use of handshake protocols. Hence, the problems associated with the clocks are avoided in these designs. Though, these designs suffer from two issues: lack of mature design tools and difficulties with respect to testing of asynchronous circuits. On the other hand, GALS design styles combine the advantages of both the asynchronous and synchronous circuits. The main motivation behind the GALS system design is to allow the synchronous blocks to operate independently with other synchronous blocks by integrating them through asynchronous communication channels. Every GALS module consists of a single synchronous block which communicates asynchronously using either FIFO or asynchronous wrappers with the synchronous block of another GALS module. Fig.1 shows the general block diagram of the GALS system and the components of an asynchronous wrapper. Every asynchronous wrapper consists of an input port, an output port and a local clock generator. The clock generator provides the clock for the particular synchronous block enclosed by that wrapper.

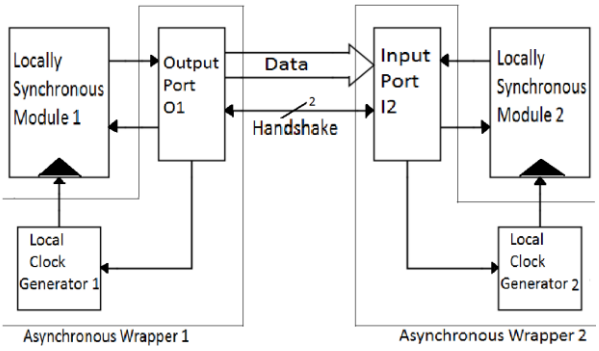

Fig 1: Block Diagram of GALS System

The output port (O1) of the first wrapper generates the request handshake for the data transfer to the input port of the second wrapper (I2) to which it attempts to send the data. The input port (I2) responds to this with an acknowledgement handshake signal indicating a ready state for the data transfer. During the handshaking process and data transfer the clocks (within the local clock generator) of the respective GALS module are stopped. This facilitates in avoiding clock distribution and synchronization between the synchronous modules, i.e. this allows each synchronous block to be operating at different/same clock frequencies.. This paper concentrates on structural testing on the different components of the asynchronous wrappers. The standard stuck-at fault model is used in order to perform the testing. The concept of fault equivalence has been used for the testing against stuck-at faults. Two types of asynchronous wrapper designs are considered in this paper: one is the Muttersbach et al wrapper [1] and the second one is the low power GALS design with stretchable clock [2].

Section II in the paper introduces the testing method. Section III describes the testing and fault equivalence applied to each of the components of the two wrapper designs. Section IV tabulates the results obtained in terms of collapse ratio and fault coverage. Finally, section V concludes the paper.

2. TESTING METHODOLOGY
This section introduces different testing methods used to test VLSI circuits. Functional testing is one of the most extensively used techniques [3]. It is often referred to as Black box testing since it does not involve testing every net in the circuit. It tests the logic functionality of the whole circuit in general. This technique requires more number of test vectors and consumes large amount of time. It is performed at the unit (gate, flip-flop) level. The other method is the structural testing method. This method requires less time compared to functional testing while maintaining the quality of the test solution. It does not check for the functionality of the circuit, rather verifies if every net in the circuit is fault free. Structural testing is divided into two types. Structural testing with

internal memory, although this method uses lesser test vectors it requires DFT (Design for Testability) circuitry, which occupies larger area. Structural testing with fault models method tests every gate in the circuit to ensure that they are free from faults of the considered fault models. This paper considers only single stuck at fault models. In this fault model, the net is fixed to a particular value either logic 1 or logic 0. If the net is fixed to logic value 1 it is called stuck-at-1 fault. If it fixed at logic value 0 it is called stuck-at-0 fault. At any point of time only one net in the circuit is either stuck at logic 1 or logic 0, hence it is referred to as single stuck-at fault model. It is assumed that if the chip is free of stuck at faults then it can be stated with 99.9% accuracy that the circuit is functionally normal. Hence it is one of the best ways to make sure that the chip is 99.9% fault free. If there are n nets, then total number of single stuck-at faults in the circuit is 2n. Structural testing is applied by considering a net in the circuit to be either stuck-at-1 (sa1) or stuck-at-0 (sa0). The primary inputs are applied such that the faulty net is driven to the opposite value of the fault considered, i.e., if a net is having a sa1 fault, the primary input should be applied in such a way that the net is driven to logic 0. The fault is then propagated to the primary output. The output obtained when the fault is present is compared with the output obtained when the fault is absent. If the result of the comparison differs then the fault is detected.

Asynchronous circuits tend to have feedback paths within the circuit. In some cases it is not possible to set the value to these paths directly from the primary inputs. In such cases, the feedback nets are broken and are given as primary inputs when necessary. This is done in order to increase the controllability of the feedback path, which is otherwise inaccessible.

Listing all possible faults and testing against them can be time consuming. Hence the concept of fault equivalence [4] is used. Two faults of a Boolean circuit are called equivalent iff they transform the circuit such that the two faulty circuits have identical output functions [4]. Fault equivalence is used to collapse the number of faults in the circuit. This gives an advantage for testing lesser number of faults yet obtaining higher fault coverage. Two important definitions [4] used throughout this paper:

Collapse ratio= (Total faults-faults removed)/Total fault
Fault coverage= 100*(Detected faults/Total faults)

Consider the circuit shown in Fig 2 as an example to demonstrate how to detect a stuck-at fault.

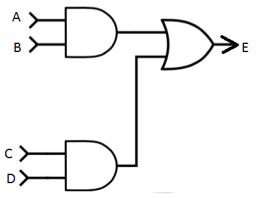

Fig 2: Example of stuck-at fault

For example, if input B is sa0, then the primary input to be applied is ABCD = 11XX. This input should generate a logic 1 at output E in absence of the sa0 fault. But in presence of the fault, the output E will be at logic 0. Hence the fault is said to be detected using the input sequence 11XX. This sequence is known as the test pattern to detect sa0 fault at input B.

3. TESTING OF WRAPPERS

3.1 Muttersbach Wrapper

This wrapper, designed by Mutterbach et al [1], is one of the most popular and widely used designs. The block diagram for the wrapper is shown in Fig 3.

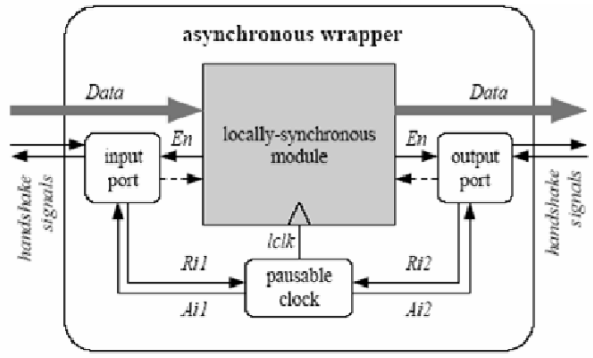

Fig 3: Mutterbach Wrapper [1]

The wrapper employs the four-phase protocol for handshaking, where each handshaking cycle comprises of four sequential events, Req+, Ack+, Req-, Ack-. The wrappers consist of I/O ports dictated by port controllers. The detailed circuit diagram of the I/O ports and the pausable clock generator are discussed next.

3.1.1 Output Port

The output port controller of the Mutterbach wrapper is shown in Fig 4. The Den is the enable signal from the locally synchronous (LS) block to the input and the output port controllers. Once the ports are triggered the output port sends a request Ri to the clock generator to pause the clock. The clock generator sends acknowledgement Ai. After the clock is paused the output port sends a request RP to the input port for data transfer. The input port pauses its clock and sends an acknowledgment signal AP to the output port, hence the data transfer commences. The acknowledgment signal from the input port makes the latch transparent to transfer the data between two synchronous blocks. This component generates the request signal RP to begin the data transfer.

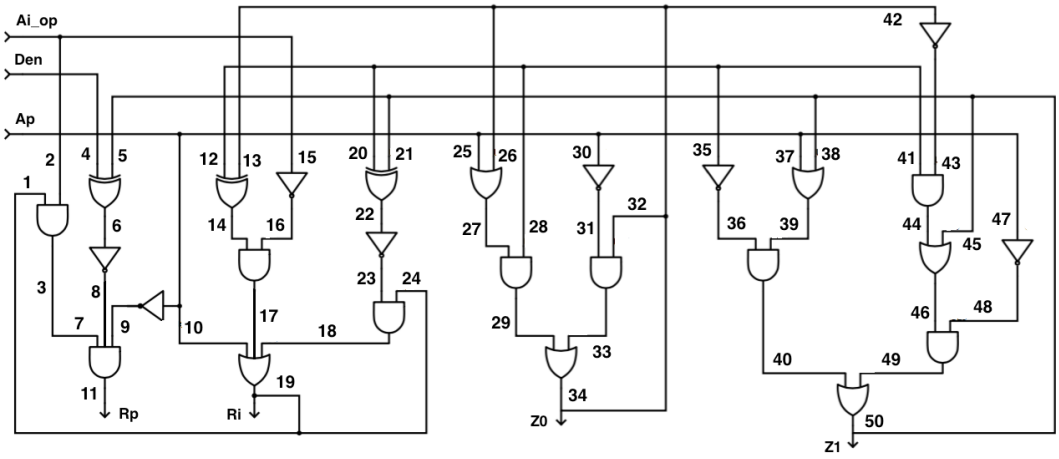

Fig 4: Output Port Controller of the Mutterbach wrapper

Let's consider net 1 from Fig 4 to be sa1. In order to detect this fault, the inputs must be set as follows: R_p=0, Den = 1, Ai = 1, Reset = 1. Also, since there are feedback loops included in this design, the initial values of these nets are unknown. Hence, these feedback loops are broken and the nets are treated as primary inputs. This helps in setting these nets to a particular value. For net 1 sa1 fault, the feedback of Ri, Ap and Z0 are broken and renamed as Ri', Ap' and Z0'. These nets are set as Ri' = 1, Ap' = 1 and Z0'=0. Now in absence of net 1 sa1 fault, the output Ri will be set to logic zero. But in presence of this fault the output will be set to logic 1. Since, the two outputs are different in absence and presence of fault, it can be said that the fault is detected. Similar analysis is performed and test patterns are obtained for all the single stuck at faults present at each gate input and output of all the circuits in this paper.

The total numbers of input and output ports are 6. Considering all the stuck at faults at all the gate inputs and outputs in the circuit, it adds up to 100 faults in the output port controller. Out of which 62 were collapsed using the fault equivalence method. The remaining 38 faults were tested using the method described in section II. The faults at location 26 and 38 cannot be detected. For these faults external hardware is required in order to access the gate inputs. The external hardware used is a 2:1 multiplexer (mux) whose select line determines the test mode. The two inputs for the 2:1 mux are AP and 1. The output is connected to the ports 30 and 47 for testing sa0 faults at 26 and 38 respectively. By including this additional hardware, both the nets can be controlled and the fault can be propagated to the output in order to detect it. The collapse ratio of this port is 0.38 and the fault coverage achieved is 100%. The maximum test vectors required to achieve 100% fault coverage is 38.

3.1.2 Input Port

This component generates the acknowledgement signal A_p and permits the data transfer. The testing has been done as mentioned in the section II. The total numbers of input and output signals are 6. Considering all the stuck at faults in the circuit, it adds up to 98 faults in the input port controller. The faults collapsed due to fault equivalence are 59. Hence the collapse ratio is 0.39 and the fault coverage achieved is 100%. The maximum test vectors required to achieve 100% fault coverage is 39.

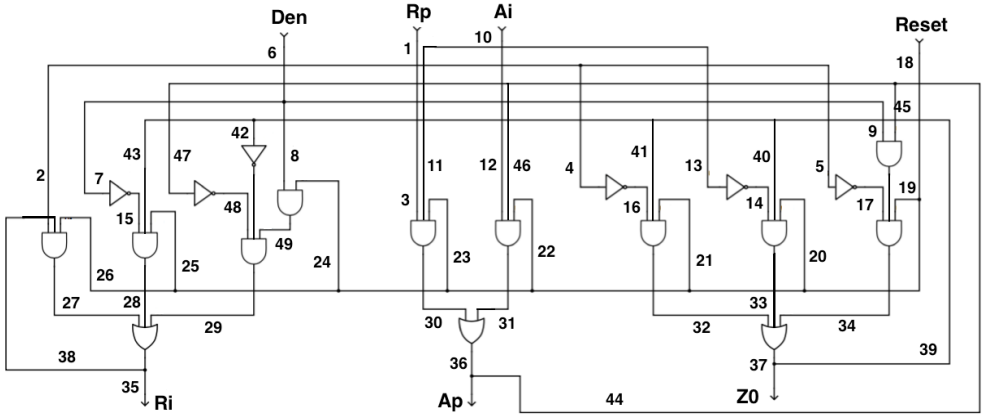

Fig 5: Input port controller of Mutterbach wrapper

3.1.3 Mutex

The mutual exclusion element arbitrates between clock and request for pause signal. The testing has been done as mentioned in the section II. The total numbers of input and output signals are 4. Considering all the stuck at faults in the circuit, it adds up to 8 faults in the mutex. The faults collapsed due to fault equivalence are 4. Hence the collapse ratio is 0.5 and the fault coverage achieved is 100%. The maximum test vectors required to achieve 100% fault coverage is 3.

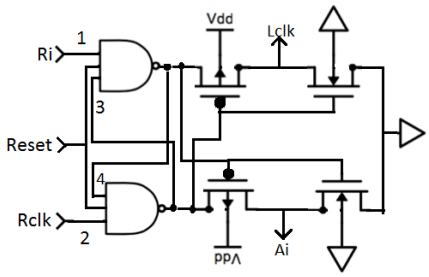

Fig 6: Mutex (pausable clock) of Mutterbach wrapper

The mutex has four transistors in its design. For these transistors, bridging [3] and stuck-open [3] faults can be considered. These transistor level faults can be mapped to the input and output faults of the mutex. After the mapping, these faults can be detected using the gate level stuck-at fault model methods. This mapping is beyond the scope of this paper; hence these types of faults have not been taken into consideration.

3.2 Low Power GALS Design with stretchable clock

This wrapper design uses a stretchable clock generator, which eliminates metastability. The design is shown in Fig 7

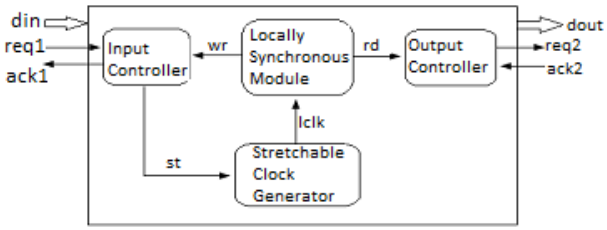

Fig 7: Design of GALS design with stretchable clock.

The output port works independently to generate request without the need to pause the clock, which makes the design less complex. However the input port must stretch the clock generator during the data transfer. The acknowledgment signal from the input port makes the latch transparent to transfer the data between two synchronous blocks. The following sections will explain the testing of each component in the wrapper.

3.2.1 Output Port

The circuit diagram for the output port controller is shown in Fig 8. The testing of this port has been performed as described in section II. The total numbers of input and output signals are 3. Considering all the stuck at faults at all the gate inputs and outputs in the circuit, it adds up to 8 faults in the output port. The number of faults collapsed is 3. The total number of input patterns to detect remaining 5 faults is 3.The sa1 fault at net 3 is detected using 2:1 mux. The collapse ratio is 0.625 and the fault coverage obtained is 100%.

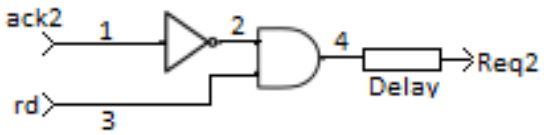

Fig 8: Output Port controller

3.2.2 Input Port

The testing of the input port controller, shown in Fig 9, has been performed as described in section II. The total numbers of input and output nets are 4. Considering all the stuck at faults in the circuit, it adds up to 30 faults in the input port controller. The number of faults collapsed is 20. The total number of input patterns to detect remaining 10 faults is 4. The collapse ratio is 0.34 and the fault coverage obtained for the input port is 100%.

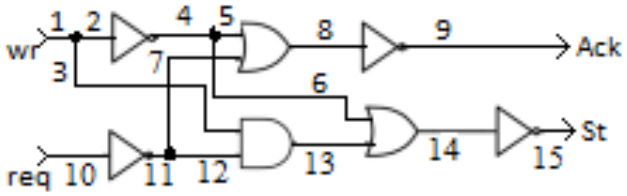

Fig 9: Input Port Controller

3.2.3 Stretchable clock

The total number of input and output nets in the stretchable clock circuit are 3. The circuit diagram for the same is shown in Fig 10. There are a total of 8 possible s-a-fault locations in the stretchable clock block. The number of faults collapsed is 3. The total number of input patterns to detect remaining 5 faults is 3. The collapse ratio is 0.625 and the fault coverage obtained is 100%.

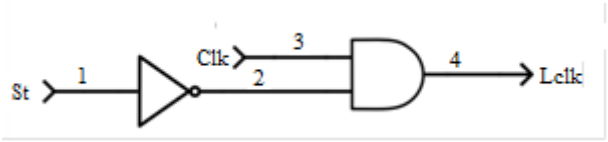

Fig 10: Stretchable Clock Generator

4. RESULTS

The two wrapper designs are implemented in Pyxis schematic in Mentor Graphics using 180nm technology. The testing for the stuck at faults is also performed using the same setup. Simulation output for one fault is shown in Fig 11 and Fig 12. A sa1 fault is introduced on net 2 of the stretchable clock generator block of the low power GALS design. The inputs given are Clk=1 and St=1. Fig 11 shows the output Lclk is logic 0 without the fault and Fig 12 shows the output Lclk is logic 1 when the fault is present. The output is different in the both cases, thus the fault is detected. All other faults are detected in the same way for both the wrapper designs.

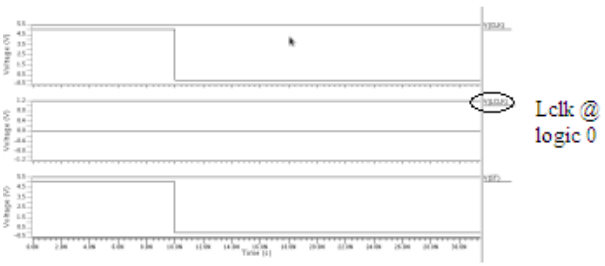

Fig 11: Simulation Result without the fault

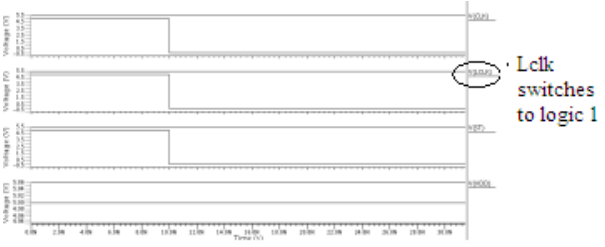

Fig 12: Simulation result with the fault

Table I formulates the testing results of Muttersbach wrapper. Table II formulates the testing results of GALS design using stretchable clock wrapper.

Table 1. Testing Results of Mutterbach Wrapper design

	Tot. no. of faults	No. of faults collapsed	Collapsed ratio	Fault Coverage (%)	Max. no. of test vectors
Output Port	100	62	0.38	100	38
Input Port	98	59	0.39	100	39
Clock Generator	8	4	0.5	100	3

Table 2. Testing Results of GALS Design using stretchable clock

	Tot. no. of faults	No. of faults collapsed	Collapsed ratio	Fault Coverage (%)	Max. no. of test vectors
Output Port	8	3	0.625	100	3
Input Port	30	20	0.34	100	4
Clock Generator	8	3	0.625	100	3

As it can be seen from these tables even though the number of faults in each block varies it is possible to detect all s-a faults in both the wrapper designs. In some cases, like the feedback paths, there is a need to use additional hardware such as a 2:1 Mux in order to detect the faults.

5. CONCLUSIONS

This paper explores the testing of GALS Designs. These designs consists of three main parts viz a synchronous block, a local clock generator and I/O port controllers. The local clock generator and I/O ports together form a wrapper designs. In this paper two different wrapper designs are explored in terms of testability.

Testing of the individual components of the wrappers has been described in detail. Structural testing with fault model and fault equivalence was applied to the two wrapper designs. The first design is the Mutterbach wrapper and the second using a stretchable clock generator with a relatively simpler design. Only single stuck at faults were considered for testing. In both wrapper designs 100% fault coverage was achieved. The collapse ratio was calculated after applying fault equivalence to the individual components. Also the maximum number of test vectors was determined in each case.

As future scope, in order to evaluate the practicality of the test approach, a full GALS design needs to be implemented and the test approached has to be applied to this design.

6. ACKNOWLEDGMENTS

The author would like to thank Snigdha Prasad, Varsha S Betagiri and Ranjitha G.

7. REFERENCES

[1] J Muttersbach, T Villiger. Fichtner, Wolfgang, 2000 Practical design of globally asynchronous locally-synchronous systems, Proceedings of Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems 52-59.

[2] C Anju, Kiriti S Pande, 2012 Low power GALS interface implementation with stretchable clocking scheme, International Journal of Computer Science Issues IJCSI, vol. 9, Issue 4, no 3.

[3] Parag Lala 1997 Digital Circuit Testing and Testability The Morgan Kaufmann Series in Computer Architecture and Design Series, Academic Press.

[4] M. Bushnell, Vishwani Agrawal, 2002 Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits, Volume 17, Springer US Publication.

[5] Frank K. Gürkaynak, Thomas Villiger, Stephan Oetiker, Norbet F elber, HubertKaeslin, Wolfgang Fichtner, 2002 A Functional Test Methodology for GALS Systems Proceedings of the 8th International Symposium on Asynchronous Circuits and Systems ASYNC.

[6] S Zeidler; M Krsti´c, 2015 A survey about testing asynchronous circuits, European Conference Circuit Theory and Design ECCTD, 1-4, 24-26

[7] L Nagy, J Koscelansky, V Stopjakova, 2014 Design of a globally asynchronous locally synchronous digital system, IEEE 12th International Conference on Emerging eLearning Technologies and Applications ICETA, 529-533.

[8] M. Krsti´c and E. Grass, 2005 BIST technique for GALS systems, Proceedings of the 8th Euromicro Conference on Digital System Design DSD, 10–16.

[9] A.Efthymiou, 2010 Initialization-Based Test Pattern Generation for Asynchronous Circuits, IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 4, 591-601.

[10] Akarsh Reddy Ravi, 2004 Globally asynchronous, locally synchronous wrapper configurations for point-to-point and multi-point data communication, M.S.Thesis, Dept. Electrical and Computer. Engineering, University of Central Florida, Orlando, Florida.

[11] D. L. Oliveira, T. Curtinhas, L. A. Faria and L. Romano, 2015 A novel asynchronous interface with pausible clock for partitioned synchronous modules, IEEE 6th Latin American Symposium, Montevideo, 1-4.